Algebraic Analysis of Branching Processes

David Delfieu, Maurice Comlan, Médésu Sogbohossou

Algebraic Analysis of Branching Processes

David Delfieu

Université de Nantes
IRCCyN
Nantes, France
david.delfieu@univ-nantes.fr

Maurice Comlan, Médésu Sogbohossou

Université d'Abomey-Calavi
LETIA
Abomey-Calavi, Bénin
comlan@hotmail.fr, sogbohossou_medesu@yahoo.fr

***Abstract*—Combinatory explosion is a limit which can be encountered when a state space exploration is driven on large specification modeled with Petri nets. Technics like unfolding have been proposed to cope with this problem. This paper presents an axiomatic model to reduce unfoldings to canonic forms which preserves conflicts.**

Keywords–Petri Nets; Unfolding; Branching process; Algebra.

I. INTRODUCTION

Petri nets are a widely used tool used to model critical real-time systems. The formal verification of properties is then based on the computation of state space [1]. But, this computation faces generally, for highly concurrent and large systems, to combinatory explosion. A major cause is the semantics of interleaving. Partial order semantics [2] have been introduced to shunt those interleavings. This work, initiated in [3], go further with the introduction of the conflict equivalence. An operator which is an abstraction of sequence and true parallelism simplifies the representation of processes, only conflicts are preserved. This approach can be used to speed up the identification of the branching processes of an unfolding. The notion of equivalence can be used to make a new type of reduction of unfoldings.

Finite prefixes of net unfoldings constitute a first transformation of the initial Petri Net (PN), where cycles have been flattened. This computation produces a process set where conflicts act as a discriminating factor. A conflict partitions a process in branching processes. An unfolding can be transformed into a set of finite branching processes. Theses processes constitute a set of acyclic graphs - several graphs can be produced when the PN contains parallelism - built with events and conditions, and structured with two operators: causality and true parallelism. An interesting particularity of an unfolding is that in spite of the loss of the concept of global marking, these processes contain enough information to reconstitute the reachable markings of the original Petri nets. In most of the cases, unfoldings are larger than the original Petri net. This is provoked essentially when values of precondition places exceed the precondition of non simple conflicts. This produces a lot of alternative conditions. In spite of that, a step has been taken forward: cycles have been broken and the conflicts have structured the nets in branching processes.

A lot of works have been proposed to improve unfolding algorithms [2][4][5][6]. Is there another way to draw on recent works about unfolding? In spite of the eventual increase of the size of the net unfoldings, the suppression of conflicts and loops has decreased its structural complexity, allowing to compute the state space and to the extract of semantics.

From a developer's point of view an unfolding can be efficiently coded by a boolean table of events. This table describes every pair to pair relation between events. This table has been the starting point of our reflection: it stresses the point that a new connector can be defined to express that a set of events belong to the same process. This connector allows to aggregate all the events of a branching process. For example, a theorem is proposed to compute all the branching processes, in canonic form, for chains of conflicts of the kind illustrated in Figure 1.

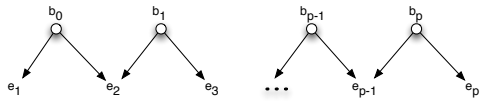

Fig 1. Chain of conflicts.

The work presented in this paper takes place in the context of combining process algebra [7][8] and Petri nets [9]. The axiomatic model of Milner's process with Calculus of Communicating Systems (CCS) is compared with the branching processes and related to other works in Section II. Then, after a brief presentation of Petri nets and unfoldings in Section III, Section IV presents our contribution with the definition of an axiomatic framework and the description of properties. The last section presents examples, in particular, illustrating a conflict equivalence.

II. RELATED WORKS

Process algebra appeared with Milner [8] on the Calculus of Communicating Systems (CCS) and the Communicating Sequential Processes (CSP) of Hoare [7] in not equivalent but similar approaches. The algebra of branching process we propose in this paper is inspired by the process algebra of Milner. *CCS* is based on two central ideas: The notion of observability and the concept of synchronized communication; *CCS* is as an abstract code that corresponds to a real program whose primitives are reduced to simple send and receive on channels. The terms (or agents) are called processes with interaction capabilities that match requests communication channels. The elements of the alphabet are observable events and concurrent systems (processes) can be specified with the use of three operators: sequence, choice, and parallelism. A main axiom of *CCS* is the rejection of distributivity of the sequence upon the choice. Let p and q be two processes, the complete process of syntax is:

$$\begin{array}{llll} \textit{Capacity} & \alpha & ::= & \bar{x} \mid x \mid \tau \\ \textit{Proces} & p & ::= & \alpha.p \mid p||q \mid p+q \mid D(\tilde{x}) \mid p\backslash x \mid 0 \end{array}$$

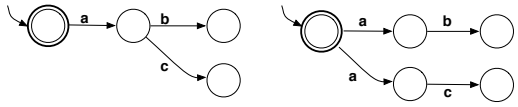

Fig 2. Milner: rejection of distributivity of sequence on choice.

Consider an observer. In the first automaton of the Figure 2, after the occurrence of the action a, he can observe either b or c. In the second automaton, the observation of a does not imply that b and c stay observable. The behavior of the two automata are not equivalent.

In *CCS*, Milner defines the observational equivalence. Two automata are observational equivalent if there are bisimular. On a algebraic point of view, the distributivity of the sequence on the choice is rejected in the equation (1):

$$a.(b+c) \not\equiv_{behaviorally} a.b+a.c \tag{1}$$

The key point of our approach is based on the fact that this distributivity is not rejected in occurrence nets. The timing of the choices in a process is essential [10]. The nodes of occurrence nets are events. An event is a fired transition of the underlying Petri net. In *CCS*, an observer observes possible futures. In occurrence nets, the observer observes arborescent past. This controversy in the theory of concurrency is an important topic of linear time versus branching time. In our model the equation (2) holds:

$$a \prec (b \perp c) \equiv (a \prec b) \perp (a \prec c) \tag{2}$$

The equation (2) is a basic axiom of our algebraic model. The equivalence relation differs then from bisimulation equivalence. This relation will be defined in the following with the definition of the canonic form of an unfolding.

Branching process does not fit with process algebra on numerous other aspects. For example, a difference can be noticed about parallelism. While unfolding keeps true parallelism, process algebra considers a parallelism of interleaving. Another difference is relative to events and conditions which are nodes of different nature in an unfolding. Conditions and events differ in term of ancestor. Every condition is produced by at most one event ancestor (none for the condition standing for m_0, the initial marking), whereas every event may have 1 or n condition ancestor(s). In *CCS*, there is no distinction between conditions and events. Moreover, conditions will be consumed defining processes as set of events.

However, a lot of works [6][10][11] have shown the interest of an algebraic formalization: it allows the study of connectives, the compositionally and facilitates reasoning (tools like [12]). Let have two Petri nets; it is questionable whether they are equivalent. In principle, they are equivalent if they are executed strictly in the same manner. This is obviously a too restrictive view they may have the same capabilities of interaction without having the same internal implementations. These work resulted to find matches (rather flexible and not strict) between nets. Mention may be made among other the occurrence net equivalence [13], the bisimulation equivalence [14], the partial order equivalence [15], or the ST-bisimulation equivalence [16]. These different equivalences are based either on the isomorphism between the unfolding of nets or on observable actions or traces of the execution of Petri nets or other criteria. This approach in this paper is weaker than a trace equivalence; it does not preserves traces but preserves conflicts. The originality of this approach is to encapsulate causality and concurrency in a new operator which "aggregates" and "abstract" events in a process. This new operator reduces the representation and accelerates the reduction process. This paper intends first, to give an algebraic model to an unfolding, and second, to establish a canonic form leading to the definition of an equivalence conflict.

III. Unfolding a Petri Net

A. Petri Net

A Petri net [9] $\mathscr{N} =< P,T,\mathscr{W} >$ is a triple with: P, a finite set of places, T, the finite set of transitions, $P \cup T$ are nodes of the net; ($P \cap T = \emptyset$), and $\mathscr{W} : (P \times T) \cup (T \times P) \longrightarrow \mathscr{N}$, the flow relation defining arcs (and their valuations) between nodes of $\mathscr{N}$.

The pre-set (resp. post-set) of a node x is denoted ${}^\bullet x = \{y \in P \cup T \mid \mathscr{W}(y,x) > 0\}$ (resp. $x^\bullet = \{y \in P \cup T \mid \mathscr{W}(x,y) > 0\}$). A marking of a Petri net $\mathscr{N}$ is a mapping $m : P \longrightarrow \mathscr{N}$. A transition $t \in T$ is said *enabled* by m iff: $\forall p \in {}^\bullet t,\ m(p) \geq \mathscr{W}(p,t)$. This is denoted: $m \xrightarrow{t}$ Firing of t leads to the new marking m' ($m \xrightarrow{t} m'$): $\forall p \in P,\ m'(p) = m(p) - \mathscr{W}(p,t) + \mathscr{W}(t,p)$. The initial marking is denoted m_0.

A Petri net is *k-bounded* iff $\forall m$, reachable from $m_0, m(p) \leq k$ (with $p \in P$). It is said *safe* when *1-bounded*. Two transitions are in a *structural conflict* when they share at least one pre-set place; a conflict is *effective* when these transitions are both enabled by a same marking. The considered Petri nets in this paper are k-bounded.

B. Unfolding

In [4], the notion of *branching process* is defined as an initial part of a run of a Petri net respecting its partial order semantics and possibly including non deterministic choices (conflicts). This net is acyclic and the largest branching process of an initially marked Petri net is called *the* unfolding of this net. Resulting net from an unfolding is a labeled occurrence net, a Petri net whose places are called *conditions* (labeled with their corresponding place name in the original net) and transitions are called *events* (labeled with their corresponding transition name in the original net).

An occurrence net [17] $\mathscr{O} =< \mathscr{B}, \mathscr{E}, \mathscr{F} >$ is a 1-valued arcs Petri net, with $\mathscr{B}$ the set of conditions, $\mathscr{E}$ the set of events, and $\mathscr{F}$ the flow relation (1-valued arcs), such that: $|{}^\bullet b| \leq 1$ ($\forall b \in \mathscr{B}$), ${}^\bullet e \neq \emptyset$ ($\forall e \in \mathscr{E}$), and $\mathscr{F}^+$ (the transitive closure of $\mathscr{F}$) is a strict order relation. This net $\mathscr{O}$ is a set of acyclic graphs. $Min(\mathscr{O}) = \{b \mid b \in \mathscr{B}, |{}^\bullet b| = 0\}$ is the minimal conditions set: the set of conditions with no ancestor can be mapped with the initial marking of the underlying Petri net. Also, $Max(\mathscr{O}) = \{x \mid x \in \mathscr{B} \cup \mathscr{E}, |x^\bullet| = 0\}$ are maximal nodes.

Three kinds of relations could be defined between the nodes of $\mathscr{O}$:

- The strict causality relation noted $\prec$: $\forall x,y \in \mathscr{B} \cup$

$\mathscr{E}$, $x \prec y$ if $(x,y) \in \mathscr{F}^+$

- The conflict relation noted #: $\forall b \in \mathscr{B}$, if $e_1, e_2 \in b^\bullet$ $(e_1 \neq e_2)$, then e_1 and e_2 are in *conflict relation*, denoted $e_1 \# e_2$ (in Figure 3.b $e_4 \# e_5$).
- The concurrency relation noted $\wr$: $\forall x, y \in \mathscr{B} \cup \mathscr{E}$ $(x \neq y)$, $x \wr y$ ssi $\neg((x \prec y) \vee (y \prec x) \vee (x \sharp y))$. (in Figure 3.b $e_2 \wr e_3$).

Remark 3.1: The transitive aspect of $\mathscr{F}^+$ implies a transitive definition of strict causality.

A set $B \subseteq \mathscr{B}$ of conditions such as $\forall b, b' \in B, b \neq b' \Rightarrow b \wr b'$ is a *cut*. Let B be a *cut* with $\forall b \in B, \nexists b' \in \mathscr{B} \backslash B,\ b \wr b'$, B is the *maximal cut*.

Definition 3.2: The unfolding $Unf_F \stackrel{\text{def}}{=} < \mathscr{O}_F, \lambda_F >$ of a marked net $< \mathscr{N}, m_0 >$, with $\mathscr{O}_F \stackrel{\text{def}}{=} < \mathscr{B}_F, \mathscr{E}_F, \mathscr{F}_F >$ an occurrence net and $\lambda_F : \mathscr{B}_F \cup \mathscr{E}_F \rightarrow \mathscr{P} \cup \mathscr{T}$ (such as $\lambda(\mathscr{B}_F) \subseteq \mathscr{P}$ and $\lambda(\mathscr{E}_F) \subseteq \mathscr{T}$) a labeling function, is given by:

1) $\forall p \in \mathscr{P}$, if $m_0(p) \neq \emptyset$, then $B_p \stackrel{\text{def}}{=} \{b \in \mathscr{B}_F \mid \lambda_F(b) = p \wedge {}^\bullet b = \emptyset\}$ and $m_0(p) = |B_p|$;
2) $\forall B_t \subseteq \mathscr{B}_F$ such as B_t is a cut, if $\exists t \in \mathscr{T}, \lambda_F(B_t) = t^\bullet \wedge |B_t| = |{}^\bullet t|$, then:
   a) $\exists! e \in \mathscr{E}_F$ such as ${}^\bullet e = B_t \wedge \lambda_F(e) = t$;
   b) if $t^\bullet \neq \emptyset$, then $B'_t \stackrel{\text{def}}{=} \{b \in \mathscr{B}_F \mid {}^\bullet b = \{e\}\}$ is as $\lambda_F(B'_t) = t^\bullet \wedge |B'_t| = |t^\bullet|$;
   c) if $t^\bullet = \emptyset$, then $B'_t \stackrel{\text{def}}{=} \{b \in \mathscr{B}_F \mid {}^\bullet b = \{e\}\}$ is as $\lambda_F(B'_t) = \emptyset \wedge |B'_t| = 1$;
3) $\forall B_t \subseteq \mathscr{B}_F$, if B_t is not a cut , then $\nexists e \in \mathscr{E}_F$ such as ${}^\bullet e = B_t$.

The definition 3.2 represents an *exhaustive* unfolding algorithm of $< \mathscr{N}, m_0 >$. In 1., the algorithm for the building of the unfolding starts with the creation of conditions corresponding to the initial marking of $< \mathscr{N}, m_0 >$ and in 2., news events are added one at a time together with their output conditions (taking into account sink transitions). In 3., the algorithm requires that any event is a possible action: there are no adding nodes to those created in item 1 and 2. The algorithm does not necessary terminate; it terminates if and only if the net $< \mathscr{N}, m_0 >$ does not have any infinite sequence. The sink transitions (ie $t \in \mathscr{T}, t^\bullet = \emptyset$) are taken into account in 2.(c).

Let be $\mathscr{E} \subset \mathscr{E}_F$. The occurrence net $\mathscr{O} \stackrel{\text{def}}{=} < \mathscr{B}, \mathscr{E}, \mathscr{F} >$ associated with $\mathscr{E}$ such as $\mathscr{B} \stackrel{\text{def}}{=} \{b \in \mathscr{B}_F \mid \exists e \in \mathscr{E}, b \in {}^\bullet e \cup e^\bullet\}$ and $\mathscr{F} \stackrel{\text{def}}{=} \{(x,y) \in \mathscr{F}_F \mid x \in \mathscr{E} \vee y \in \mathscr{E}\}$ is *a prefix* of $\mathscr{O}_F$ if $Min(\mathscr{O}) = Min(\mathscr{O}_F)$. By extension, $Unf \stackrel{\text{def}}{=} < \mathscr{O}, \lambda >$ (with λ, the restriction of λ_F to $\mathscr{B} \cup \mathscr{E}$) is a prefix of unfolding Unf_F.

It should be noted that, according to the implementation, the names (the elements in the sets $\mathscr{E}$ and $\mathscr{B}$) given to nodes in the same unfolding can be different. A name can be independently chosen in an implementation using a tree formed by its causal predecessors and the name of the corresponding nodes in $\mathscr{N}$ [4].

Definition 3.3: A causal net $\mathscr{C}$ is an occurrence net $\mathscr{C} \stackrel{\text{def}}{=} < \mathscr{B}, \mathscr{E}, \mathscr{F} >$ such as:

1) $\forall e \in \mathscr{E} : e^\bullet \neq \emptyset \ \wedge \ {}^\bullet e \neq \emptyset$;
2) $\forall b \in \mathscr{B} : |b^\bullet| \leq 1 \ \wedge \ |{}^\bullet b| \leq 1$.

Definition 3.4: $\mathscr{P}_i = (\mathscr{C}_i, \lambda_F)$ is a *process* of $< \mathscr{N}, m_0 >$ iff: $\mathscr{C}_i \stackrel{\text{def}}{=} < \mathscr{B}_i, \mathscr{E}_i, \mathscr{F}_i >$ is a causal net and $\lambda : \mathscr{B}_i \cup \mathscr{E}_i \rightarrow P \cup T$ is a labeling fonction such as:

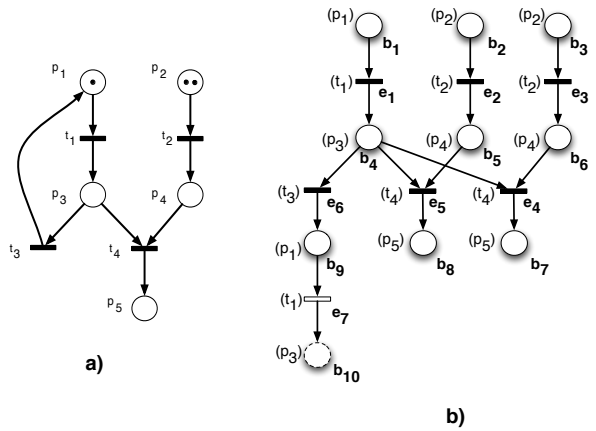

Fig 3. a) Petri net, b) Unfolding.

1) $\mathscr{B}_i \subseteq \mathscr{B}_F$ *and* $\mathscr{E}_i \subseteq \mathscr{E}_F$
2) $\lambda_F(\mathscr{B}_i) \subseteq P$ *and* $\lambda_F(\mathscr{E}_i) \subseteq T$;
3) $\lambda_F({}^\bullet e) = {}^\bullet \lambda_F(e)$ *and* $\lambda_F(e^\bullet) = \lambda_F(e)^\bullet$
4) $\forall e_i \in \mathscr{E}_i, \ \forall p \in P : \ \mathscr{W}(p, \lambda_F(e)) = |\lambda^{-1}(p) \cap {}^\bullet e| \ \wedge \ \mathscr{W}(\lambda_F(e), p) = |\lambda^{-1}(p) \cap e^\bullet|$
5) If $p \in Min(P) \Rightarrow \exists b \in \mathscr{B}_i : \ {}^\bullet b = \emptyset \ \wedge \ \lambda_F(b) = p$

$Max(\mathscr{C}_i)$ is the *state* of $\mathscr{N}$. $Min(\mathscr{C}_i)$ and $Max(\mathscr{C}_i)$ are maximum cuts. Generally, any maximal cut $B \subseteq \mathscr{B}_i$ corresponds to a reachable marking m of $< \mathscr{N}, m_0 >$ such as $\forall p \in \mathscr{P}, m(p) = |B_p|$ avec $B_p = \{b \in B \mid \lambda(b) = p\}$.

The *local configuration* of an event e is defined by: $[e] \stackrel{\text{def}}{=} \{e' \mid e' \prec e\} \cup \{e\}$ and is a process. For example of unfolding in Figure 3.b: $[e_4] \stackrel{\text{def}}{=} \{e_1, e_3, e_4\}$.

The conflits in a unfolding derive from the fact that there is an reachable marking (a cut in an unfolding) such as two or many transitions of a labelled net $< \mathscr{N}, m_0 >$ are *enabled* and the firing of one transition disable other. Whence the proposition:

Proposition 3.5: Let be $e_1, e_2 \in \mathscr{E}_F$. If $e_1 \perp e_2$, then there $\exists (e'_1, e'_2) \in [e_1] \times [e_2]$ such as ${}^\bullet e'_1 \cap {}^\bullet e'_2 \neq \emptyset$ et ${}^\bullet e'_1 \cup {}^\bullet e'_2$ is a cut.

IV. Branching Process Algebra

The previous section showed how unfolding exhibits causal nets and conflicts. Otherwise, every couple of events which are not bounded by a causal relation or the same conflict set are in concurrency. Then, an unfolding allows to build a $2D$-table making explicit every binary relations between events. Practically, this table establishes the relations of causality and exclusion. If a binary relation is not explicit in the table, it means that the couple of events are in a concurrency relation.

Let $\mathscr{E}\mathscr{B} = \mathscr{E} \cup \mathscr{B}$ a finite alphabet, composed of the events and the conditions generated by the unfolding. The event table (produced by the unfolding) defines for every couple in $\mathscr{E}\mathscr{B}$ either a causality relation $\mathscr{C}$, either a concurrency relation $\mathscr{I}$ or an exclusive relation $\mathscr{X}$. These sets of binary relations dot

not intersect and the following expressions can be deduced:

$$Unf/_{\mathscr{X}} = \mathscr{C} \cup \mathscr{I} \quad (3)$$
$$Unf/_{\mathscr{C}} = \mathscr{X} \cup \mathscr{I} \quad (4)$$
$$Unf/_{\mathscr{I}} = \mathscr{C} \cup \mathscr{X} \quad (5)$$

To illustrate these relation sets, the negation operator noted $\neg$ can be introduced. Then, the equations (3),(4),(5) leads to (6),(7),(8):

$$\neg((e_1,e_2) \in \mathscr{I}) \Leftrightarrow (e_1,e_2) \in \mathscr{C} \cup \mathscr{X} \quad (6)$$
$$\neg((e_1,e_2) \in \mathscr{C}) \Leftrightarrow (e_1,e_2) \in \mathscr{I} \cup \mathscr{X} \quad (7)$$
$$\neg((e_1,e_2) \in \mathscr{X}) \Leftrightarrow (e_1,e_2) \in \mathscr{C} \cup \mathscr{I} \quad (8)$$

The equation (8) expresses that if two events are not in conflict they are in the same branching process. Let us now define the union of binary relations $\mathscr{C}$ and $\mathscr{I}$: $\mathscr{P} = \mathscr{C} \cup \mathscr{I}$ For every couple $(e_1,e_2) \in \mathscr{P}$, either (e_1,e_2) are in causality or in concurrency: $\mathscr{P}$ is the union of every branching process of an unfolding.

a) Example: Let us consider an unfolding (left part) on the Figure 4 and the table T (right part) which is its representation: In Figure 4, the table T contains 7 causal

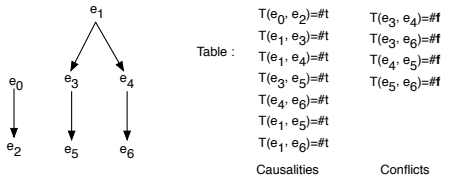

Fig 4. Unfolding.

relations and 4 conflict relations. (e_0,e_4) is not (negation) in the table, expresses that e_0 and e_4 are concurrent. Moreover, if two events are not in conflict (consider e_0 and e_6): (e_0,e_6) is not a key of the table, (e_0,e_6) are in concurrency and thus, those events belongs to the same branching process.

A. Definition of the Algebra

The starting point of this work is based on the fact that the logical negation operator articulates the relation between two sets: the process set $\mathscr{P}$, and the exclusion set $\mathscr{X}$. As mentioned in Section IV, $\mathscr{C}$, $\mathscr{I}$ *and* , $\mathscr{P}$ does not intersect, then semantically, if a couple of events is not in a relation of exclusion (noted $\perp$), the events are in $\mathscr{P}$. $\mathscr{P}$ contains binary relations between events that are in branching process.

To express that events are in the same branching process, a new operator noted $\oplus$ is introduced. An algebra describing branching process can be defined as follow:

$$\{\mathscr{U}, \prec\ ,\ \wr\ ,\ \perp\ ,\oplus,\ \neg\}$$

Let us note; $* = \oplus, \prec,$ or $\perp$, $\#t$ the void process, and $\#f$ the false process. Here is the formal signature of the language:

- $\forall e \in \mathscr{E}\mathscr{B}, e \in \mathscr{U}, \#t \in \mathscr{U}, \#f \in \mathscr{U}$
- $\forall e \in \mathscr{U}, \neg e \in \mathscr{U}$
- $\forall (e_1,e_2) \in \mathscr{U}^2, e_1 * e_2 \in \mathscr{U}$

Properties, neutral/absorbing elements, distributivities and semi-distributivities have been defined in [3]. However, let us now just recall the definitions (equations (9),(10),(11),(12)):

1) Causality: $\mathscr{C}$ is the set of all the causalities between every elements of $\mathscr{E}\mathscr{B}$. $e_1 \prec e_2$ if e_1 is in the local configuration of e_2:

$$e_1 \prec e_2 \text{ if } e_1 \in [e_2] \quad (9)$$

2) Exclusion: $\mathscr{X}$ is the set of all the exclusion relations between every elements of $\mathscr{E}\mathscr{B}$. Two events are in exclusion *iff* they are either in direct conflict, either it exists a conflict at any level with an ancestor:

$$e_1 \perp e_2 \equiv ((\bullet e_1 \cap \bullet e_2 \neq \emptyset) \text{ or } (\exists e_i, e_i \prec e_2 \text{ and } e_1 \perp e_i)) \quad (10)$$

3) Concurrency: $\mathscr{I}$ is the set of every couple of element of $\mathscr{E}\mathscr{B}$ in concurrency. e_1 and e_2 are in concurrency if the occurrence of one is independent of the occurrence of the other. So, $e_1 \wr e_2$ iff e_1 and e_2 are neither in causality neither in exclusion.

$$e_1 \wr e_2 \equiv \neg((e_1 \perp e_2) \text{ or } (e_1 \prec e_2) \text{ or } (e_2 \prec e_1)) \quad (11)$$

4) Process: $\oplus$ aggregates events in one process. Two events e_1 and e_2 are in the same process if e_1 causes e_2 or if e_1 is concurrent with e_2:

$$e_1 \oplus e_2 \equiv (e_1 \prec e_2) \text{ or } (e_2 \prec e_1) \text{ or } (e_1 \wr e_2) \quad (12)$$

This operator constitutes an abstraction which hides in a black box causalities and concurrency. The meaning of this operator is similar to the linear connector $\oplus$ of MILL [18]. It allows to aggregates resources. But, in the context of unfolding, events or conditions are unique and then they cannot be counted. Thus, this operator is here idempotent.

The expression $e_1 \oplus e_2$ defines that e_1 and e_2 are in the same process.

Note that $(\oplus\ e_1\ e_2\ ...\ e_{n-1}\ e_n)$ will abbreviate $(e_1 \oplus\ e_2 \oplus e_3 \oplus ...e_{n-1} \oplus\ e_n)$

B. Axioms

The following axioms stem directly from previous assumptions and definitions made upon the algebraic model:

Axiom 4.1 (Distributivity of $\prec$):

$$e \prec (e_1 \perp e_2) \equiv_{def} (e \prec e_1) \perp (e \prec e_2)$$

The first axiom constitutes the basis of our approach. As discussed in the Section II, on the contrary of *CCS*, e is distributed onto two expressions, giving alternative process.

Axiom 4.2 (Definition of $\oplus$):

$$e_1 \oplus e_2 \equiv_{def} (e_1 \prec e_2) \perp (e_2 \prec e_1) \perp (e_1 \wr e_2)$$

$\oplus$ aggregates two elements in a process. Two elements are in a process if they are concurrent or in a causality relation.

Axiom 4.3 ($\prec$):

$$e_1 \prec e_2 \equiv_{def} \neg e_1 \perp (e_1 \oplus e_2)$$

A causality can be expressed by two processes in exclusion: either $\neg e_1$: e_1 has not occurred either $e_1 \oplus e_2$: e_1 and e_2 within the same process.

Axiom 4.4 (Duality between $\oplus$ and $\perp$):

$$e_1 \oplus e_2 \equiv_{def} e_1 \neg\perp e_2 \qquad e_1 \neg\oplus e_2 \equiv_{def} e_1 \perp e_2$$

This axiom comes from the introduction of the operator $\neg$ discussed in the beginning of the Section IV. It expresses that $\mathscr{P}$ and $\mathscr{X}$ are complementary sets.

Axiom 4.5 (Exclusion):

$$e_1 \perp e_2 \equiv_{def} (\neg e_1 \oplus e_2) \perp (e_1 \oplus \neg e_2)$$

The fifth axiom expresses that a conflict can be considered as two processes in conflict

Axiom 4.6 (Distributivities):

- $\prec, \perp, \oplus$ are distributive over $\wr$.
- $\prec, \oplus, \wr$ are distributive over $\perp$ (axiom 4.3).
- $\perp, \wr$ are distributive over $\perp$ and $\oplus$.

The distributivities over $\perp$ are used in the transformation of an expression in the canonic form. The other distributivities will be used in the reduction process.

C. Canonic Form

1) Definition: The definition of the canonic form allows to define an equivalence called a "conflict equivalence".

Definition 4.1: A canonic process is a formula expressed on elements of $\mathscr{EB}$ and with the operators $\oplus$, $\perp$ ordered by an alphanumeric sort on the name of its symbol.

Theorem 4.2 (Canonical form): Let us consider an unfolding U, this form can be reduced in the following form:

$$U = (\perp\ P_1\ P_2\ ...\ P_n) \qquad \text{where} \quad P_i = (\oplus\ e_{i_1} ...\ e_{i_n})$$

This form is canonic and exhibits every processes P_i of the unfolding.

Proof: In an unfolding every causality ($\prec$) and every partial order ($\wr$) can be reduced in $\oplus$ by deduction rules Modus Ponens (MP, MP_1, MP_2), Simplification rule (S) and *Par* (see Section IV-C2). Moreover, $\oplus$ and $\perp$ are mutually distributive, so $\perp$ can be factorized in every sub-formula to reach the higher level of the formula. In fine, an alphanumeric sort on symbols of the processes can be applied to assure the unicity of the form. ■

This canonic form preserves conflicts, let us now define a *conflict equivalence*:

Definition 4.3 (Conflict Equivalence): Let us U_1, U_2 unfoldings of Petri nets:

$$U_1 \approx_{conf} U_2 \qquad \text{iff they have the same canonic form.}$$

Remark 4.4: A process is an aggregate set of events where $\prec$ and $\wr$ are hidden. This equivalence is lower than a trace equivalence: each process P_i is an abstraction of a set of traces.

2) Derivation Rules: This section gives a set of rules which transform branching processes toward a canonical form. Theses transformations preserve conflicts whereas $\prec$ and $\wr$ are transformed in $\oplus$.

Let us note b a condition, e an event and E a well formed formula on the algebra. Theses rules allow to reduct process:

1) Modus Ponens:

$$\frac{\vdash \oplus b... \quad \vdash \oplus b... \prec e}{\vdash e}\ MP_1 \qquad \frac{\vdash e \quad \vdash e \prec \oplus b...}{\vdash \oplus\ e\ b...}\ MP_2$$

Where $\oplus b...$ stands for the general form for $\oplus b_1 b_2 ...\ b_n$. MP_1 expresses that $b...$ are consumed by the causality, whereas, in MP_2 e stays in the conclusion.

2) Dual form: $\dfrac{\vdash \neg e_1 \quad \vdash e_1 \prec e_2}{\vdash \neg e_1 \oplus \neg e_2}\ MP'$

3) Simplification: $\dfrac{\vdash \neg e_1 \oplus E}{\vdash E}\ S_1 \qquad \dfrac{\vdash \oplus b...\ E}{\vdash E}\ S_2$

Those rules are applied, *in fine*, to clear not pertinent informations in the process. S_1 rule is applied, to clear the negations whereas S_2 is applied to clear the conditions which have not been consumed.

4) Reduction of $\wr$: $\dfrac{\vdash e_1 \wr e_2}{\vdash e_1 \oplus e_2}\ Par$

This rule corresponds to the definition of $\oplus$
These rules have been defined to lead to a canonic form.

D. Theorems

The properties of operators (definition, axioms and distributivites) allow to define theorems which are congruences w.r.t the operators of Section IV-A (proofs have been already stated in [3]).

Theorem 4.5 (Conflict):

$$e_1 \prec (e_2 \perp e_3) \equiv (e_1 \prec (e_2 \oplus \neg e_3)) \perp (e_1 \prec (\neg e_2 \oplus e_3))$$

This theorem expresses how to develop a conflict and the following theorem allows to reduce processes:

Theorem 4.6 (Absorption): Let E, F some processes: $E \perp (E \oplus F) \equiv E \oplus F$

1) Chain of conflicts: This section presents a theorem which computes the branching process in canonic form of a chain of conflict illustrated in Figure 5.

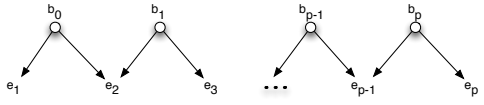

Fig 5. Chain of conflicts.

The axiomatic representation of the unfolding is:

$$U = ((\oplus\ b_0\ b_1\ ...\ (b_0 \prec (e_1 \perp e_2))(b_1 \prec (e_2 \perp e_3))...)$$

After some steps of reduction $(MP + S)$:

$$U = (e_1 \perp e_2 \perp ... \perp\ e_p)$$

Let us now consider the following conventions: Let us note:

- $l^1 = (e_1, e_2, ...e_n)$, $l^2 = (e_2, ...e_n)$
- l_i the i^{th} element of a list l.
- If e_i is an element of the list l, let us note $indice(e_i)$ the position of e_i in l.

Remark 4.7: In the list of event constituting a chain of conflict ($l = (e_1, e_2, ...e_n)$), for every event e_i, the next (resp. previous) event in the same branching process is e_{i+2} or e_{i+3} (resp. e_{i-2} or e_{i-3})

The next definition defines two processes U_n and V_n which are aggregation of events where the possible successor of an event e_i is either $l_{(indice(e_i)+2)}$ either $l_{(indice(e_i)+3)}$.

Definition 4.8: Let us consider that $n <= p$ (p: index of the last event implied in the chain of conflict):

$$\begin{cases} U_0 = e_1 \\ U_n^1 = l_{n+2}^1 \oplus U_{n+2}^2 \\ U_n^2 = l_{n+3}^1 \oplus U_{n+3}^2 \\ U_n = U_n^1 \ \oplus U_n^2 \end{cases} \qquad \begin{cases} V_0 = e_2 \\ V_n^1 = l_{n+2}^2 \oplus V_{n+2}^2 \\ V_n^2 = l_{n+3}^2 \oplus V_{n+3}^2 \\ V_n = V_n^1 \ \oplus V_n^2 \end{cases}$$

U_n: processes beginning by e_1 V_n: processes beginning by e_2

Theorem 4.9: The canonic form of a chain of conflict C is $U_n \oplus V_n$:

$$(e_1 \perp \ e_2 \perp \ ... \perp \ e_p) \equiv U_n \oplus V_n$$

V. Examples

A. Example 1

The Figure 6 gives a Petri net which represents a chain of conflicts and its unfolding. The unfolding gives a table of

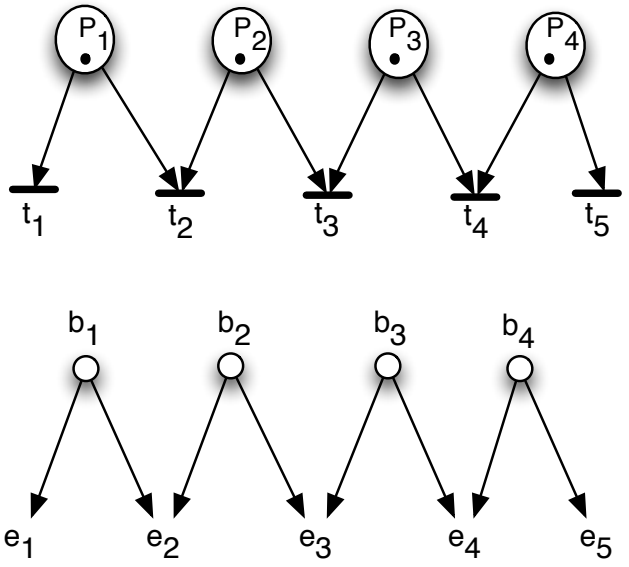

Fig 6. PN and unfolding of a chain of conflicts.

binary relations on events (see Section IV) which is represented by the following algebraic expression U_2:

$$U_1 = (\oplus \ b_1 \ b_2 \ b_3 \ b_4 \ b_5 \ (b_1 \prec (e_1 \perp e_2)) \ (b_2 \prec (e_2 \perp e_3)) \ ...)$$

After some steps of reduction ($MP + S$), U_1 becomes:

$$(e_1 \perp \ e_2 \perp \ e_3 \perp \ e_4 \perp \ e_5) \tag{13}$$

The theorem (T 4.9) allows to compute from (13) its following canonic form:

$$(\perp (\oplus \ e_1 \ e_3 \ e_5)(\oplus \ e_1 \ e_4 \ )(\oplus \ e_2 \ e_4)(\oplus \ e_2 \ e_5))$$

B. Example 2

Let us consider the following Unfolding of the Figure 7. The table computed which leads to the following algebraic

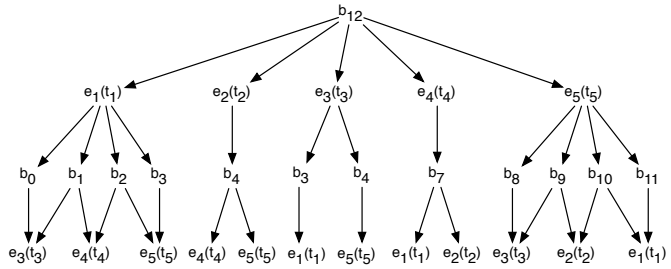

Fig 7. U_2.

expression U_2:

$$\begin{aligned} U_2 \quad = (\oplus \ b_{12} \quad & (b_{12} \prec (e_1 \perp \ e_2 \perp \ e_3 \perp \ e_4 \perp \ e_5)) \\ & (e_1 \prec (\oplus \ b_0 \ b_1 \ b_2 \ b_3))(e_2 \prec b_4)(e_3 \prec (\oplus b_5 \ b_6))(e_4 \prec b_7) \\ & (e_5 \prec (\oplus b_8 \ b_9 \ b_{10} \ b_{11}))((\oplus \ b_0 \ b_1) \prec e_3) \ ((\oplus \ b_1 \ b_2) \prec e_4) \\ & ((\oplus \ b_2 \ b_3) \prec e_5) \ (b_4 \prec (\perp e_4 \ e_5))(b_5 \prec e_1) \ (b_6 \prec e_5) \\ & (b_7 \prec (\perp e_1 \ e_2)) \ ((\oplus \ b_8 \ b_9) \prec e_3) \ ((\oplus \ b_9 \ b_{10}) \prec e_2) \\ & ((\oplus \ b_{10} \ b_{11}) \prec e_1)) \end{aligned} \tag{14}$$

Let us note P the aggregation of the five first lines of the equation. (14) becomes:

$$U_2 \quad = (\oplus \ b_{12} \quad (b_{12} \prec (\perp \ e_1 \ e_2 \ e_3 \ e_4 \ e_5)) \ P \tag{15}$$

Rules $MP1$, MP_2 and theorem 1, reduce (15) in:

$$U_2 \quad = (\perp (\oplus \ e_1 \ P) \ (\oplus \ e_2 \ P) \ (\oplus \ e_3 \ P) \ (\oplus \ e_4 \ P) \ (\oplus \ e_5 \ P) \ )$$

Distributivity of *perp*:

$$\begin{aligned} U_2 \quad = (\oplus \quad & (\perp (\oplus \ e_1 \ b_0 \ b_1 \ b_2 \ b_3)(\oplus \ e_2 \ b_4)(\oplus \ e_3 \ b_5 \ b_6) \ (\oplus \ e_4 \ b_7) \\ & (\oplus \ e_5 \ b_8 \ b_9 \ b_{10} \ b_{11})) \ ((\oplus \ b_0 \ b_1) \prec e_3) \ ((\oplus \ b_1 \ b_2) \prec e_4) \\ & ((\oplus \ b_2 \ b_3) \prec e_5) \ (b_4 \prec (\perp e_4 \ e_5))(b_5 \prec e_1) \ (b_6 \prec e_5) \\ & (b_7 \prec (\perp e_1 \ e_2))((\oplus \ b_8 \ b_9) \prec e_3) \ ((\oplus \ b_9 \ b_{10}) \prec e_2) \\ & ((\oplus \ b_{10} \ b_{11}) \prec e_1)) \end{aligned}$$

Distributivity of $\perp$ and MP_1:

$$\begin{aligned} U_2 \quad = (\perp \quad & (\oplus \ e_1 \ e_3 \ e_5 \ b_1 \ b_2)(\oplus \ e_1 \ e_4 \ b_0 \ b_3)(\oplus \ e_2 \ e_4)(\oplus \ e_2 \ e_5) \\ & (\oplus \ e_3 \ e_1)(\oplus \ e_3 \ e_5)(\oplus \ e_4 \ e_1)(\oplus \ e_4 \ e_2)(\oplus \ e_5 \ e_1 \ e_3 \ b_9 \ b_{10}) \\ & (\oplus \ e_5 \ e_2 \ b_8 \ b_{11})) \end{aligned}$$

Theorem 2 (absorption of $(\oplus \ e_3 \ e_1)$ and $(\oplus \ e_3 \ e_5)$ in $(\oplus \ e_1 \ e_3 \ e_5 \ b_1 \ b_2)$, idempotency of $\perp$:

$$\begin{aligned} U_2 \quad = (\perp \quad & (\oplus \ e_1 \ e_3 \ e_5 \ b_1 \ b_2)(\oplus \ e_1 \ e_4 \ b_0 \ b_3)(\oplus \ e_2 \ e_4)(\oplus \ e_2 \ e_5) \\ & (\oplus \ e_4 \ e_1)(\oplus \ e_5 \ e_1 \ e_3 \ b_9 \ b_{10})(\oplus \ e_5 \ e_2 \ b_8 \ b_{11})) \end{aligned}$$

Rules of simplification S_1 and S_2 and theorem 2:

$$U_2 = (\perp (\oplus \ e_1 \ e_3 \ e_5)(\oplus \ e_1 \ e_4)(\oplus \ e_2 \ e_4)(\oplus \ e_2 \ e_5))$$

The two unfoldings of example 1 and 2 have the same canonic form, they are *conflict-equivalent*: $U_1 \approx_{conf} U_2$

1) Reasoning about processes: Let us consider all the process p of $U_2 : (\oplus\ e_1\ e_3\ e_5), (\oplus\ e_1\ e_4), ...$

- $\forall p \in U_2$ whenever e_3 is present, e_1 is present.
- $\forall p \in U_2, \neg e_3 \perp (e_1 \oplus\ e_3 \oplus e_5)$
  This is the algebraic definition of $\prec$. Finally from this chain of conflicts, the following causality can be deduced:
  $$e_3 \prec (e_1 \oplus e_5) \tag{16}$$
- A similar reasoning can be made:
  $$\forall p \in U_2, \neg(e_1 \oplus e_5) \perp (e_1 \oplus\ e_3 \oplus e_5)$$
  This is the algebraic definition of:
  $$(e_1 \oplus e_5) \prec e_3 \tag{17}$$
  (16) and (17) expresses that there is a *strong link* between e_3 and the process $(e_1 \oplus e_5)$ but $\prec$ is no well suited to encompass this relation. Theses two processes are like "intricated".
- In the same manner:

$$\begin{aligned} \neg e_2 \perp (e_2 \oplus e_4) \perp (e_2 \oplus e_5) \quad &\equiv_{dist} \quad \neg e_2 \perp (e_2 \oplus (e_4 \perp e_5)) \\ &\equiv_{def} \quad e_2 \prec (e_4 \perp e_5) \end{aligned} \tag{18}$$

e_2 leads to a conflict

$$\begin{aligned} \neg e_1 \perp ((\oplus e_1 e_3 e_5) \perp (e_1 \oplus e_4) \quad &\equiv_{dist} \quad \neg e_1 \perp (e_1 \oplus ((e_3 \oplus e_5) \perp e_4)) \\ &\equiv_{def} \quad e_1 \prec ((e_3\ \oplus\ e_5) \perp e_4) \end{aligned} \tag{19}$$

Equations (18) and (19) show that e_1 and e_2 transform the chain of conflict in a unique conflict. New relations between events or processes can be introduced:

- Alliance relation: e_1, e_3 and e_5 are in "an alliance relation". Every event of this set is enforced by the occurrence of the other events: $e_1 \oplus e_3$ enforces e_5, $e_1 \oplus e_5$ enforces e_3 and $e_3 \oplus e_5$ enforces e_1.
- Intrication: the occurrence of e_3 forces $e_1 \oplus e_5$ and reciprocally $e_1 \oplus e_5$ forces e_3.
- Resolving conflicts (liberation):
  - e_1 resolves 3 conflicts on 4 (as e_2, e_4 and e_5)
  - e_3 resolves every conflicts.

Semantically, e_3 can be identified as an important event in the chain. Moreover $(\oplus e_1\ e_3\ e_5)$ is a process aggregated with "associated events". This chain of conflict can be seen as two causalities in conflicts: $(e_1 \prec (e_4 \perp (e_3 \oplus e_5)))\ \perp\ (e_2 \prec (e_4 \perp e_5))$

2) Example 3 (Cash dispenser): Let us consider the cash dispenser of the Figure 8, its unfolding in canonical form is:

$U3 = (\perp (\oplus\ \mathit{Consult\ EnterCode\ OKcode\ GetConsult})$
$(\oplus\ \mathit{Consult\ EnterCode\ OKcode\ Getcash})$
$(\oplus\ \mathit{Consult\ EnterCode\ BadCode})$
$(\oplus\ \mathit{Cash\ EnterCode\ OKcode\ Getcash})$
$(\oplus\ \mathit{Cash\ EnterCode\ OKcode\ GetConsult})$
$(\oplus\ \mathit{Cash\ EnterCode\ BadCode}))$

This expression enlightens that *GetCash* and *BadCode* are neither in the same process.

VI. Conclusion and Future Work

This work is a first attempt to present an axiomatic framework to the analyze of the processes issued of an unfolding. From a set of axioms, distributivities and derivation rules, theorems have been established and a reduction process can lead to a canonic form The unfolding process, definitions, theorems and reduction rules have been coded in LISP[19] with a package named PLT/Redex[3][12]. This canonic form assets an equivalence conflicts ($\equiv_{conf}$) between unfoldings and then Petri nets.

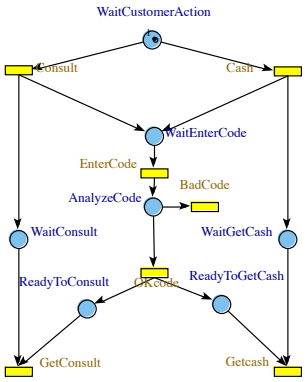

Fig 8. *Cash dispenser.*

Several perspectives are into progress. First, news theorems have to be established allowing to speed up the procedure of canonic reduction and to extend extraction of knowledge on relationship between events. Different kinds of relationship between events have to be defined and formalized: Alliance relation, Intrication, etc. Moreover, as already outlined in the last part of the example section, algebraic reasoning can raise semantic informations about events from the canonic form. Another perspective is to extend this approach to Petri nets with inhibitor and drain arcs.

References

[1] B. Berthomieu and F. Vernadat, "State class constructions for branching analysis of time petri nets," in TACAS, volume 2619 of LNCS. Springer Verlag, 2003, pp. 442–457.

[2] J. Esparza and K. Heljanko, "Unfoldings - a partial-order approach to model checking," EATCS Monographs in Theoretical Computer Science, 2008.

[3] D. Delfieu and M. Sogbohossou, "An algebra for branching processes," in Control, Decision and Information Technologies (CoDIT), 2013 International Conference on, May 2013, pp. 625–634.

[4] J. Engelfriet, "Branching processes of petri nets," Acta Informatica, vol. 28, no. 6, 1991, pp. 575–591.

[5] J. Esparza, S. Römer, and W. Vogler, An Improvement of McMillan's Unfolding Algorithm. Mit Press, 1996.

[6] McMillan and Kenneth, "Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits," in Computer Aided Verification. Springer, 1993, pp. 164–177.

[7] C. A. R. Hoare, Communicating sequential processes. Prentice-hall Englewood Cliffs, 1985, vol. 178.

[8] R. Milner, Communication and concurrency. Prentice-hall Englewood Cliffs, 1989.

[9] C. A. Petri, "Communication with automata," PhD thesis, Institut fuer Instrumentelle Mathematik, 1962.

[10] R. Glabbeek and F. Vaandrager, "Petri net models for algebraic theories of concurrency," in PARLE Parallel Architectures and Languages Europe, ser. Lecture Notes in Computer Science, J. Bakker, A. Nijman, and P. Treleaven, Eds. Springer Berlin Heidelberg, 1987, vol. 259, pp. 224–242.

[11] E. Best, R. Devillers, and M. Koutny, "The box algebra=petri nets+process expressions," Information and Computation, vol. 178, no. 1, 2002, pp. 44 – 100.

[12] M. Felleisen, R. Findler, and M. Flatt, Semantics Engineering With PLT Redex. Mit Press, 2009.

[13] M. Nielsen, G. Plotkin, and G. Winskel, "Petri nets, event structures and domains," in T. Theor. Comp. Sci., vol. 13(1), 1981, pp. 89–118.

[14] J. Baeten, J. Bergstra, and J. Klop, "An operational semantics for process algebra," in CWI Report CSR8522, 1985.

[15] G. Boudol and I. Castellani, "On the semantics of concurrency: partial orders and transitions systems," in Rapports de Recherche No 550, INRIA, Centre Sophia Antipolis, 1986.

[16] V. Glaabeek and F. Vaandrager, "Petri nets for algebraic theories of concurrency," in CWI Report SC-R87, 1987.

[17] T. Chatain and C. Jard, "Complete finite prefixes of symbolic unfoldings of safe time petri nets," in Petri Nets and Other Models of Concurrency - ICATPN 2006, ser. Lecture Notes in Computer Science, S. Donatelli and P. Thiagarajan, Eds. Springer Berlin Heidelberg, 2006, vol. 4024, pp. 125–145. [Online]. Available: http://dx.doi.org/10.1007/11767589

[18] J.-Y. Girard, "Linear logic," Theoretical computer science, vol. 50, no. 1, 1987, pp. 1–101.

[19] G. L. Steele, Common LISP: the language. Digital press, 1990.